RL-TR-94-147
Final Technical Report
August 1994

ENHANCING THE ARCHITECTURAL CHARACTERIZATION OF PARALLEL PROTO

AD-A285 678

Syracuse University

Daniel J. Pease

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

RL-TR-94-147 has been reviewed and is approved for publication.

APPROVED: Milissa M. Benincasa

MILISSA M. BENINCASA
Project Engineer

FOR THE COMMANDER: John A. Graniero

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE August 1994	3. REPORT TYPE AND DATES COVERED Final Oct 92 - Oct 93

4. TITLE AND SUBTITLE ENHANCING THE ARCHITECTURAL CHARACTERIZATION OF PARALLEL PROTO	5. FUNDING NUMBERS C - F30602-92-C-0062 PE - 63728F PR - 2527 TA - 03 WU - PA
6. AUTHOR(S) Daniel J. Pease	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Syracuse University Department of Electrical and Computer Engineering Center for Science and Technology Syracuse NY 13244	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory (C3CB) 525 Brooks Road Griffiss AFB NY 13441-4505	10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-94-147

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Milissa M. Benincasa/C3CB/(315) 330-7650

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The objective of the work presented in this report was to determine the feasability of integrating the architectural database of the Parallel Assessment Window System (PAWS) into the database of the Parallel Proto (PPROTO) tool. PAWS and PPROTO are Rome Laboratory developed tools. The PAWS tool is an experimental system for performing machine evaluation and comparisons. The tool provides the user with the ability to predict the performance of an application on a number of selected parallel architectures. PPROTO is a software engineering environment that allows a user to validate and analyze the requirements of parallel and distributed systems using rapid prototyping techniques. The benefit of being able to integrate the PAWS database into the PPROTO database is that high level performance prediction can be performed at the requirements phase of the software engineering lifecycle. This would provide users the capability of determining which parallel architecture would be most suitable for their particular problem domain.

14. SUBJECT TERMS Performance assessment, Parallel processing, Software tools, Software engineering, High performance computers	15. NUMBER OF PAGES 28 16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

Final Report

for Rome Laboratory Project:

Enhancing the Architectural Characterization of Parallel Proto

1.0 Executive S[illegible]ary

Short-term and long-term technical tasks were defined for the project. The short-term tasks and long-term tasks have been completed on schedule.

The capabilities of the Parallel Proto (PPROTO) tool have been studied by the Syracuse University (SU) Team and are well understood. Data elements from the Parallel Assessment Window System (PAWS) tool data base have been identified for usage in characterizing the MultiMax and the Touchstone parallel architectures within the existing capabilities of PPROTO. This characterization is restricted by the limitations of the present version of PPROTO.

The creation of data members for PPROTO architectural characterization can be automated, but cannot be linked to the simulation and visualization. Several recommendations have been developed that can allow more accurate representations of parallel architectures in PPROTO.

In this effort an attempt was made to establish an interface between the architectural data base of PAWS and PPROTO. The objective of this work was to enhance the architectural characterization of PPROTO. Due to the present form of PPROTO the automated interface could not be developed.

The Syracuse University Team was led by Dr. Daniel Pease, the Principal Investigator. Mr. Mikki was responsible for Visualization aspects of the architectural characterization, Mr. Foudil-Bey was responsible for the interface with PPROTO, and Mr. Zerrouki was responsible for the interface to the PAWS Architectural Data Base.

The following tasks have been completed under this effort:

1. Design, test and refinement of the Architectural Visualization Interface to PPROTO.

2. Design, test and refinement of the Interface to PPROTO.

3. Design, test and revise refinement to PAWS architectural data base.

There are three additional sections to this report:

- Task Status,
- Technical Results, and
- Summary of Specific Recommendations

2.0 Task Status

1. Study the existing Architectural Visualization in PPROTO and propose an interface to Architectural Data from PAWS.

Schedule:	Start by:	May 27, 1992
	Complete by:	September 27, 1992
Status:	Completed:	September 27, 1992

2. Design and implement an Architectural Visualization Interface to PPROTO.

Schedule:	Start by:	September 27, 1992
	Complete by:	March 27, 1993
Status:	Completed:	March 27, 1993

3. Test and refine the Architectural Visualization Interface to PPROTO.

Schedule:	Start by:	March 27, 1993
	Complete by:	May 27, 1993
Status:	Completed:	May 27, 1993

Parallel Proto Interface

1. Study the existing Architectural Interface in PPROTO and propose an interface to Architectural Data from PAWS.

Schedule:	Start by:	May 27, 1992
	Complete by:	September 27, 1992
Status:	Completed:	September 27, 1992

2. Design and implement an Interface to PPROTO.

Schedule:	Start by:	September 27, 1992
	Complete by:	March 27, 1993
Status:	Completed:	March 27, 1993

3. Test and refine the Interface to PPROTO.

Schedule:	Start by:	March 27, 1993
	Complete by:	May 27, 1993
Status:	Completed:	May 27, 1993

2.0 Task Status (Continued)

PAWS Interface Refinement

1. Study the existing Architectural Interface to PAWS.

Schedule: Start by:	May 27, 1992
Complete by:	September 27, 1992
Status: Completed:	September 27, 1992

2. Design a refinement to PAWS architectural data base so it can be Interfaced to PPROTO.

Schedule: Start by:	September 27, 1992
Complete by:	March 27, 1993
Status: Completed:	March 27, 1993

3. Test and revise refinement to PAWS architectural data base.

Schedule: Start by:	March 27, 1993
Complete by:	May 27, 1993
Status: Completed:	May 27, 1993

3.0 Technical Results and Recommendations

There are three fundamental technical tasks that have been completed under this effort. Specifically:

1. Design, test and refinement of the Architectural Visualization Interface to PPROTO.

2. Design, test and refinement of the Interface to PPROTO.

3. Design, test and revise refinement to PAWS architectural data base.

The results of each of these tasks are a set of recommendations for further modifications to PPROTO and PAWS.

3.1 Design, Test and Refine the Architectural Visualization Interface to PPROTO.

Architectural Visualization is one of the key components of PPROTO . In PPROTO, a parallel computer system is defined by drawing a graphical picture of the architecture. This first task consisted of several parts aimed at determining if a new architecture could be added to PPROTO without manual intervention. Currently, in PPROTO, the user has to draw the architecture and define its high level characteristics manually. In order to automate this process the following steps were taken:

1. Manually Use PAWS Architectural Data to Create Two PPROTO Architectural Visualizations

2. Analyze User Interface to PPROTO Architectural Visualization

3. Design an automated interface that would take PAWS Architectural Characterization data and create a PPROTO Visualization

It was felt at the onset of this effort that the third step was not possible due to the design structure of the PPROTO tool.

3.1.1 Manually Use PAWS Architectural Data to Create Two PPROTO Architectural Visualizations

In order to understand Architectural Visualization in PPROTO, two real-world parallel architectures were developed for PPROTO using the PAWS architectural data. The Intel Touchstone Hypercube and the MultiMax 320A MIMD parallel architectures were utilized. Once the parallel architectures had been determined a set of queries were defined that allowed accessibility to the detailed architectural data base information in PAWS. These queries included the following specific requests for information that are presently entered manually into the PPROTO Architectural Characterization:

Using a distribution for different types of mathematical operations to get a single time needed for PPROTO. For example, the PPROTO add time was determined from the sum of 50% of the integer add time, 30% of the floating point add time, 10% of the long integer add time, and 10% of the double precision add time.

The number of Processors, number of memories and the number of interconnection paths between processors/memories.

Using a distribution for different types of memory access and interconnect communication operations to get a single time needed for PPROTO. For example, the PPROTO interconnect access time was determined from the sum of the cache access time, 10% of the coherency evaluation time, 10% of the main memory access time, 1% of the secondary storage access time, and 1% of the paging time.

Once this information was obtained it was then used to manually draw the two parallel architectures in PPROTO. These were completed and delivered to Rome Laboratory (RL) in the five month report. An example of the hypercube parallel architecture entered into PPROTO is shown in Figure 1.

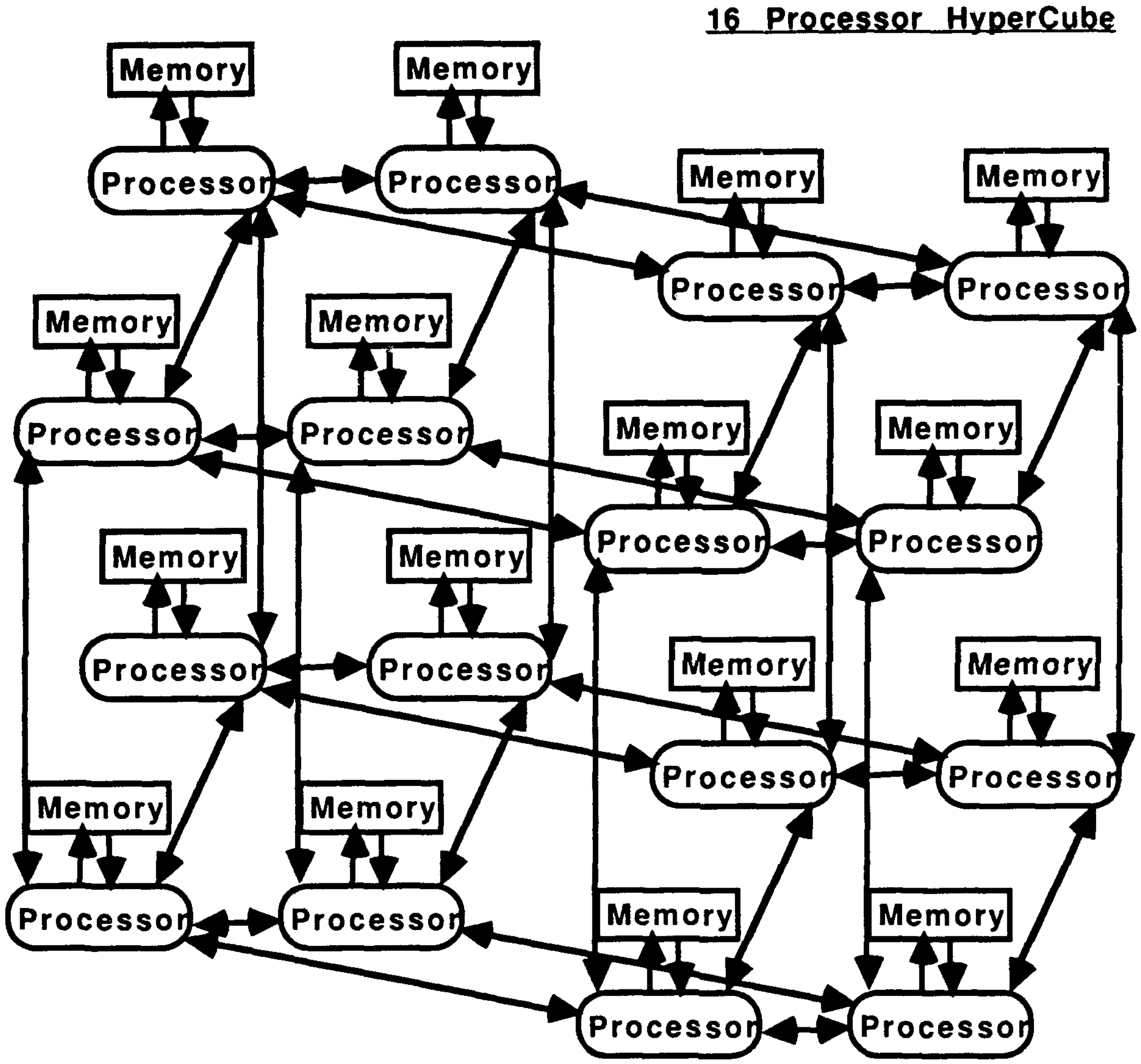


Figure 1 - An Example of a 16 Processor Touchstone Drawn in PPROTO

3.1.2 Analyze User Interface to Architectural Visualization

An architecture, or parallel computer system is input into PPROTO manually by graphically constructing the system with the elements provided by PPROTO Architecture tool.

PPROTO provides a 'flat' two-dimensional view of systems. PPROTO provides the following elements with which to create an architecture:

a simple processor,
an interconnection bus,
a processor to bus connection,
a memory to bus connection, and
a memory block.

The user creates the element by selecting from the menu of five pieces defined above. Once an option has been selected the new element is displayed and is automatically named by PPROTO with an internal name. This internal name is used by PPROTO to uniquely identify each element in the architecture. All of the architectural information is stored in the ONTOS data base by PPROTO. The user can move elements around to make the architecture visually understandable. However as the number of elements gets large this is difficult in a two dimensional environment.

The architecture is then characterized by a table of execution times of the fundamental processor operations and a memory access delay. There are 33 data elements included in PPROTO. They include the following:

<u>OPERATION TIMES</u>

Add/Subtract - The time it takes a processor to add/subtract two values that are stored in registers in the processor,

Multiply - The time it takes a processor to multiply two values that are stored in registers in the processor,

Divide - The time it takes a processor to divide two values that are stored in registers in the processor,

Boolean Operations (including Compare) - The time it takes a processor to logically relate two values that are stored in registers in the processor.

Complex Operations - The time it takes a processor to perform a complex operation on a set of values that are stored in registers in the processor, (for example, a value raised to a power).

<u>ACCESS TIMES</u>

Bus Access Time - The time it takes to get a data value into a register from memory. This is treated as a delay time which PPROTO adds to the processing time.

The above values must be set by the user or a unit default time is used. The unit default time is a single unit of the smallest time increment being used to characterize an architecture in PPROTO. PPROTO does not use absolute time, only relative time is used with regard to unit time. For example, if the smallest operation time in the architecture is 10 nanoseconds for an add, the unit time would be assumed to be 10 nanoseconds. To enter the multiply time of 30 nano seconds requires the user to enter a time of 3 (3 times the unit time). It is up to the user to determine the unit time, and to convert all of the operation times for the architecture into times relative to the unit time. If the user does not know a time the default unit time is used. This is not good measure since the unit time refers to the fastest time. These values are presently entered manually from the PPROTO screen AFTER the element has been drawn. It is important to note the relationship between these parameters and the ability of PPROTO to characterize an architecture. If the user does not know the time for an operation such as exponentiation (which is a complex activity many times longer than something like addition) the time that PPROTO will use will be the unit time. This is not realistic since it makes the architecture look as though it is much faster than it actually is. The following values have to be provided for an architecture:

Architecture	PPROTO Values
Processor	Operation Times (Fixed Integer)
Interconnect	Bus Access Time (Fixed Integer)
Memory	Nothing (Imbedded in Access)

Each of these elements is stored in the ONTOS data base that is created for each architecture. The ONTOS data base is a third party product that the developers of PPROTO are using to manage the large amounts of data that make up PPROTO. The ONTOS data base is a relational data base that allows data and the relationships between data to be stored and accessed without the PPROTO developers having to actually manage the physical storage and organization of the data. The ONTOS data base is accessed by a collection of callable functions which perform different storage and retrieval functions. The characteristics of the data base are:

1. The Operation Time is stored as a link list.

2. The Bus Access Time is stored as a separate entity in data base structure.

3. The remainder of the data base elements are derived values generated internally by PPROTO. These include an extensive set of coordinates and characteristics of each architectural element.

The user interface provides no mechanism for linking a part of an existing architecture and its associated data set with a new architecture, (i.e. no Cut and Paste across architectures.)

3.1.3 Design an Automated Interface that takes PAWS Architectural Characterization Data and Creates a PPROTO Visualization

This task consisted of three parts which attempted to interface the PAWS Architecture Characterization Data Base with PPROTO. The task included the following steps:

1. Create a Query to the PAWS Architecture Characterization data base to provide a set of data compatible with PPROTO.
2. Automatically create an architecture in PPROTO that is based on data returned by querying the PAWS Architecture Characterization data base.
3. Automatically modify an existing architecture in PPROTO that has been stored in the ONTOS data base, by inserting the data returned from querying the PAWS Architecture Characterization Data Base .

A query of the PAWS Architecture Characterization data base was successfully completed. This query did not take advantage of the plethora of information currently available in the PAWS Architecture Characterization Data Base. For example, in order to obtain the add time, for input into PPROTO, the integer add time, the floating point add time, the long integer add time, and the double precision add time were accessed.

The internal PPROTO data that is stored in the ONTOS Data Base could not be externally generated. This task was unsuccessful because the data that had been extracted from the PAWS Architecture Characterization data base could not be utilized directly by the user interface in PPROTO. The user interface in PPROTO provides the capability of drawing the architecture. An attempt was made to create a file that contained all of the data as a script in the correct order, so that it could be manually input to PPROTO, but it was not possible to activate the script will PPROTO was running.

The next approach tried to directly insert the data into the ONTOS data base. This was done by over-writing a set of data that had been created manually with PPROTO. This worked successfully for small architectures of less then 20 total elements, but for larger architectures there were linkage pointer problems with ONTOS.

If an architecture could have been created automatically in PPROTO the following steps would have been required:

1. Extract the data required to characterize the architecture in PPROTO from the PAWS Architecture Characterization Data Base.
2. From the data set extracted, create a new architectural data base in ONTOS for PPROTO, inserting the data from PAWS.
3. From the extracted data create a visualization of the new architecture.

A file containing the external information needed to create the architecture in PPROTO was successfully accomplished, but steps 2 and 3 above could not be accomplished without major code revisions to PPROTO. The interface to the architectural drawing part of PPROTO could be modified to allow the user to use a script which would contain the definition of each element in the architecture, the elements position on the screen, and the data needed to characterize the element.

Another possible modification would involve PPROTO performing an overwrite on an existing architecture containing data that is currently stored in the ONTOS data base. PPROTO would read from a file rather than deriving the architecture from the screen drawing program. PPROTO would have complete control of the data and the ONTOS data base to ensure integrity and correct operation.

3.1.3.1 PAWS Data Extraction

The query interface to the PAWS architectural data base was used with a small modification to extract the data needed to create an architecture within PPROTO. The modification made to the PAWS architectural data base required a way to detect the number of processors, memories and interconnects associated with an architecture and establish a screen position for them. This had not been an original requirement of the PAWS architectural data base. The specific data required included:

Architecture Naming Information - A unique name for the system being modeled, such as Touchstone 16 for a 16 processor Touchstone hypercube architecture.

Operation Times - The time required to do adds, multiplies and other processor operations.

Access Time - The time required to access data from memory and bring it to the processor.

Netlist of Processor, Interconnect, and Memory Connections - The interconnections that exist between each piece of the architecture. these are pair-wise point-to-point connections.

Number of Processors - The number of processor blocks.

Number of Interconnects - The number of bus blocks.

Number of Memories - The number of memory blocks.

The operation times and access time were average values computed from more detailed PAWS data.

The netlist was a modification to the query interface of the PAWS architectural data base. This was accomplished by creating a processor to interconnect list and an interconnect to memory list. These were then combined into the netlist that was required for PPROTO.

3.1.3.2 Create Externally an Architecture in PPROTO

The process PPROTO uses to create the architecture data base in ONTOS was traced. The most of the ONTOS calls that were made could be identified but some of the internal PPROTO parameters that were passed in the calls could not. Attempts to use the same variable names were unsuccessful.

3.1.3.3 Create Externally an Architecture Graph

In PPROTO, the architecture graphs are described by a special graphical language called Swordfish. Swordfish is a special language created by the developers of PPROTO to simplify the generation of graphics in the X-Windows environment. Swordfish has a set of graphical functions which have simple calls and parameter sets which produce complete X-Window graphical figures. Swordfish is used in PPROTO for all of the visual/graphical drawings that make up the user interface in PPROTO. The Swordfish operations could not be accessed properly, due to undefined local parameters and unknown Swordfish function calls. Swordfish is good for internal development, but it excludes needed support for external construction of graphs.

Recommendation #1: Create the visual graph of the architecture manually, then have PPROTO generate all the data it needs to create an architecture. Next, create a file that contains values for each element in the set of data that PPROTO created, then the user can modify the file with all of the data needed to *correctly specify the architecture. The file is then returned* to PPROTO, and then PPROTO will load it into the ONTOS data base, thus giving the graph the parameters needed to represent the architecture.

3.2 Design, Test and Refine the Interface to PPROTO.

The first task completed was an in depth study of how PPROTO stores and architecture.

Parallel Proto Architecture Study

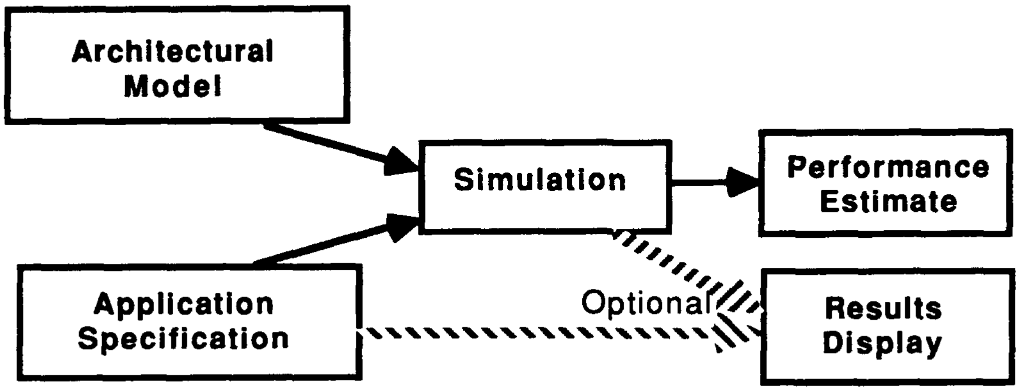


Figure 1 - Present Parallel Proto System

As shown in Figure 1, there are three primary parts to PPROTO design:

Architectural Model - A simple model indicating the relationship between processors and primary memory in the computer system being targeted for execution.

Application Specification - A graphically based logical model of the software including all of the major portions of the functional specification . These are primarily data flow diagrams. A major part of this is an object-oriented data base of all primary data structures in the software.

Simulation - The structure of the data flow which represents how the application software specification is mapped to the architectural model and how the execution time is analyzed. The execution simulation is targeted to the parallel system specified, so parallel mapping is done.

An attempt was made to add a new interface to PPROTO as shown in Figure 2. This interface attempted to incorporate the PAWS architectural data base into PPROTO.

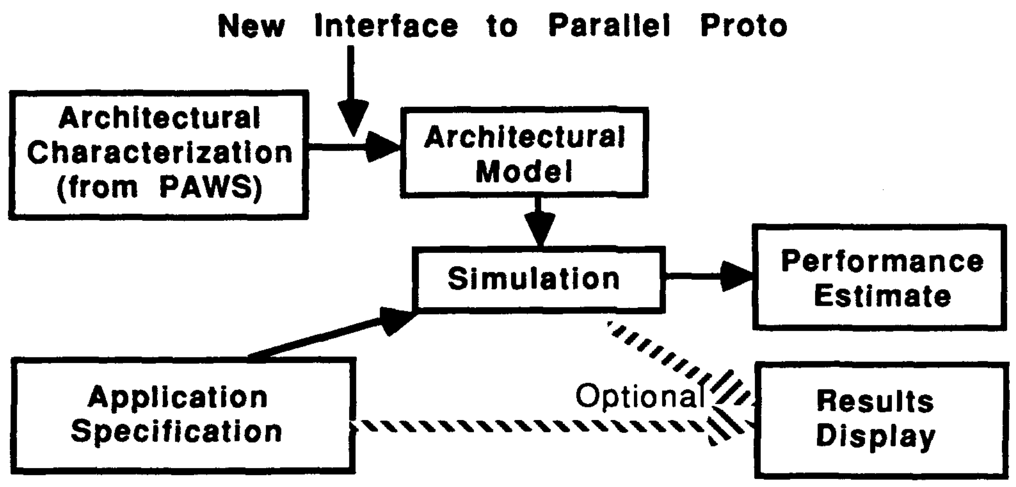

New Interface to Parallel Proto
Architectural Characterization (from PAWS)
Architectural Model
Simulation
Performance Estimate
Application Specification
Optional
Results Display

Figure 2 - Proposed Parallel Proto System Revision

Four sub tasks were defined to do this. Together these sub tasks evaluated whether PPROTO could be made more usable for Rome Laboratory. The sub tasks were:

1. Evaluate whether data of increased detail like that in the PAWS data base could be utilized in the PPROTO simulations.

2. Identify how an architecture model is stored in PPROTO.

3. Interface the PAWS architectural data base to PPROTO.

4. Test the new architectural characterization with a test application.

PAWS takes a low level approach to architecture characterization. Data has been accumulated on architecture parameters which potentially can affect performance in a parallel system. This accumulated data was incorporated into a data base where it is accessible by a user and can be utilized by the other tools in the PAWS system.

3.2.1. Evaluate whether Data of Increased Detail like that in the PAWS Data Base can be used in the PPROTO Simulations

It was determined that the following data was required in order to create an architecture in PPROTO:

Architecture naming information,

Operation times,

Access time,

Netlist of processor, interconnect, and memory connections,

Number and location of processors,

Number and location of interconnects, and

Number and location of memories.

Locations of processors, interconnects, and memories, netlist, and some of the times had to be derived by utilizing an algorithm on the PAWS data. The netlist was a modification to the query interface of the PAWS architectural data base. This was accomplished by creating a processor to interconnect list and an interconnect to memory list. These were then combined into the netlist that was required for PPROTO. The remaining values could be extracted directly from PAWS.

The following parameters available in the PAWS data base are the most significant ones that could not be utilized in the PPROTO Architecture Characterization:

Execution Time Variations by Data Type (including integer, float, long and double precision values had to be combined into a single value)

Hierarchical Memory information -

- Cache information (Line Size, Replacement Policy, Size, Speed,...)
- Main Memory information (Page Size, Replacement Policy, Size, Speed, Coherency Management,...)
- Extended Memory information (Storage Unit, Swap Policy, Replacement Policy, Size, Speed, Coherency Management,...)
- Secondary Storage information (Storage Unit, Swap Policy, Size, Speed, Latency,...)
- Memory Management Hardware

Register information - (Integer registers, Vector registers, Pointer Registers,...)

Operating System Penalties (including synchronization penalty, system function execution times, ...)

Variations in Communication Path Rates (based on path length, contention, traffic density, differences between different hierarchical levels, ...)

Interconnect Routing Policies (including time for blocking resolution, broadcast costs, actual configuration and switching times, ...)

Recommendation #2: Each data item should be considered by PPROTO designers to determine if it can be used within the framework of performance analysis within PPROTO. It is particularly important that the addition of a hierarchical memory model be made. Most of the existing parallel systems have a hierarchical memory system. The communication and memory access costs are the most significant part of the penalty in expected parallel performance. The performance of the memory system has been shown to be 50% - 80% of the total time of execution in parallel systems.

3.2.2. Identify How the Architecture Model is Stored in PPROTO

PPROTO characterizes an architecture using the following parameters (these were previously defined in PAWS Data Extraction paragraph in Section 3.1.3):

Architectural Identifiers,

Number and locations of Processors,

Number and locations of Interconnects,

Number and locations of Memories,

Processor Operation Times,

Interconnect Bus Access Delay, and

Arcs that connect memories, interconnects and processors.

Each of these elements is stored in the ONTOS data base that is created for each architecture. The characteristics of the data base are:

1. All numerical data is stored as integers.
2. Operation Times are stored as a link list.
3. Bus Access Time is stored as a separate entity in the data base structure.
4. There are data base elements that are locally derived values generated internally by PPROTO.
5. Locations are absolute Swordfish graph coordinates.

Recommendation #3: How the PPROTO data base is designed should be left up to the developers of PPROTO. PPROTO developers because of some of the limitations of ONTOS might want to consider other data base managers that have more streamlined interfaces. This effort did not consider or evaluate other data bases, but it was felt that there is a great deal of overhead associated with the ONTOS data base. But as long as PPROTO establishes an external data base interface, it should maintain internal control over the data base structure and operation.

3.2.3. Interface the PAWS Architectural Data Base to PPROTO

The goal of this task was to take the extracted PAWS data needed to characterize an architecture and insert it into an existing PPROTO architecture. This was accomplished for small architectures, architectures with less than 20 elements. But for large architectures over 20 elements (this included processors, interconnects, and memories) this technique failed.

The approach used to accomplish this task included the following steps:

1. Trace the PPROTO code that was used to edit the values that characterize the processor operation times and the access delay.

2. Isolate this code and remove all the user interactions, except those needed to identify the architecture.

3. Open the file of extracted PAWS data and insert each element as though it were an edit done manually by the user.

4. When all the data is inserted save the architecture and end the edit session.

The code could be executed externally from PPROTO, but it could not be activated from within the architecture structure of PPROTO Attempts at trying this method did not produce an error code message from the ONTOS data base, but several attempts corrupted the data base.

Recommendation #4: How the PPROTO data base is designed should be left up to the developers of PPROTO. PPROTO should provide a data list of each architecture after its structure is initially entered. This list should be generated by PPROTO and contain exactly what is needed to properly set up the architectural data that is needed internally. By having PPROTO generate the list it guarantees that the correct data in the correct format will be used. The user can take this list and edit it to insert the data required and return it to PPROTO. PPROTO can then take the data and insert it into the ONTOS data base. This guarantees data integrity and the proper management of the internal structure of the architectural information for PPROTO. It also guarantees that the ONTOS data base is correct.

3.2.4. Test the New Architectural Characterization with a Test Application

Four test cases were attempted:

Test Case 1 - Eight Processor MultiMax
Test Case 2 - Eight Processor Hypercube
Test Case 3 - Twenty Processor MultiMax
Test Case 4 - Thirty-Two Processor Hypercube

The following steps were processed for each test:

Step 1 - Create the architecture manually in PPROTO.

Step 2 - Extract the data from the PAWS architecture data base as previously describe in PAWS Data Extraction paragraph in section 3.1.3. This data contained the parameters needed by PPROTO for the selected architecture.

Step 3 - Insert the extracted PAWS data into the PPROTO architecture by copying it element by element into the data structure.

In all four cases the creation of the architecture, extraction of PAWS data and insertion ran to completion. In test cases 1 and 2, for small architectures, the resultant PPROTO architecture was usable and contained reasonable data. For test cases three and four, for large architectures, the PPROTO architectures that were created would not execute and PPROTO failed. The error message was traced to a pointer error in the PPROTO Data Base.

3.3 Design, Test and Revise Refinements to PAWS Architectural Data Base

In order to use the PAWS data two special operations had to be performed while the data was being extracted. These operations are:

1. Scaling - PPROTO requires that all architecture values must be integers. The integer values are multiples of the smallest unit of time for an operation being modeled within PPROTO. The user has to determine this by calculating the operation times for all architecture values, finding the smallest and then scaling every one of the operations to the smallest value.

2. Value Derivation - Parameters that did not exist explicitly in the PAWS architectural data base, had to be calculated.

Scaling was actually done after extraction and derivation. The resultant values were stored in a file for later use in PPROTO.

Scaling

All the values were searched and a minimum scale unit was determined that could reasonably be divided into all of the extracted and derived values. This minimum scale unit value was then used to convert all of the values into integers within the range allowed by PPROTO.

Value Derivation

Four sets of values had to be derived from PAWS data since they were not directly available as parameters in the PAWS architectural data base. These values included:

1. Some operation times were average values were computed from more detailed PAWS data.

2. The access time was an average value computed from more detailed PAWS data.

3. The netlist was a modification to the query interface of the PAWS architectural data base.

4. The graphical locations of processors, interconnects and memories.

A set of simple algebraic relations were developed for each of these sets of values that used available data. These relations have been defined in PAWS Data Extraction paragraph in Section 3.1.3 and in the other parts of this section.

Operation Times

It was assumed that all operations used register data and that the mix of types were 50% integer times, 30% floating point times, 10% long integer times, and 10% double precision times.

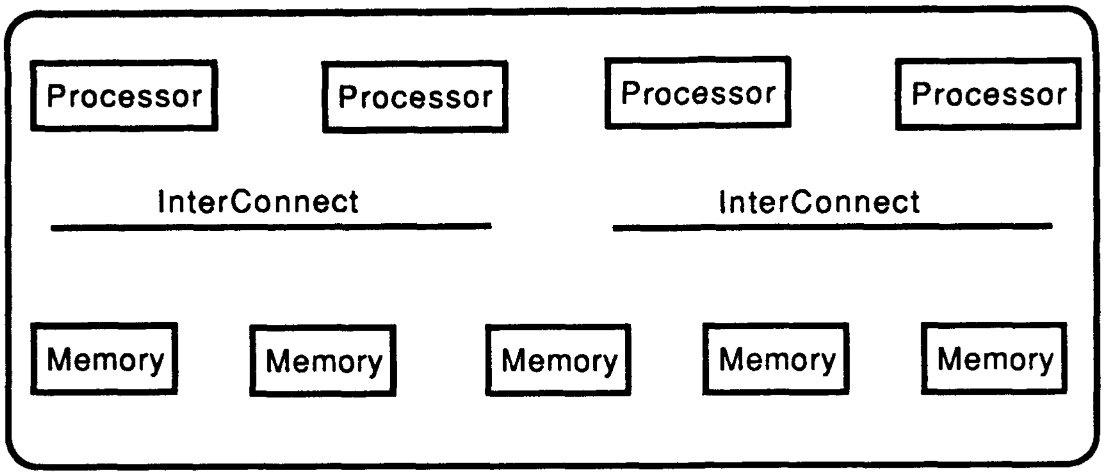
Access Time

No secondary memory penalties were included, and no cache speedups were allowed. Main memory access time was defined as an integer.

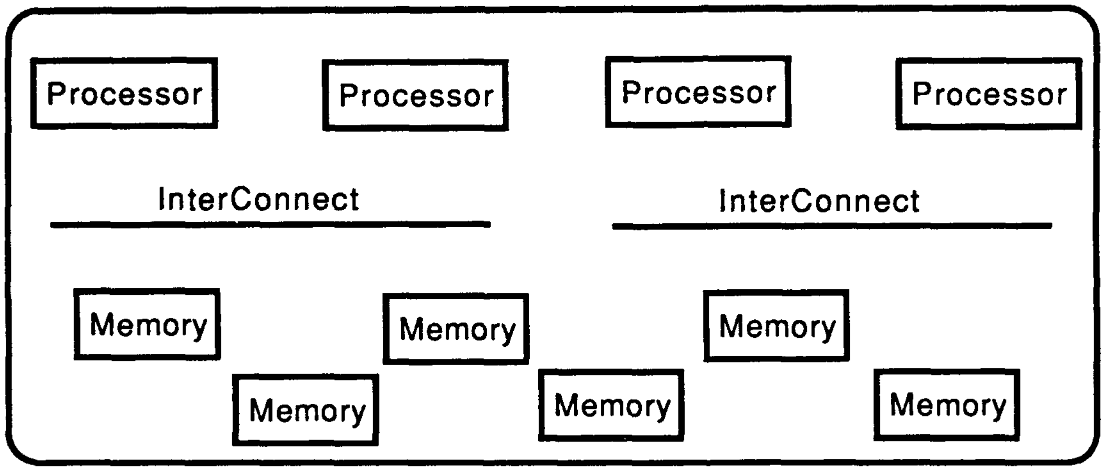
Netlist

This was accomplished by creating a processor to interconnect list and an interconnect to memory list. The two lists were then combined into the netlist that was required for PPROTO.

Three Level Graph Positioning Example



Three Level Graph Positioning with Stagger



Location

A three level grid was established that distributed processors evenly over the top level, interconnects evenly over the middle level, and memories evenly over the bottom level. If there was not enough room a staggered two level arrangement was used for each level of elements. This is shown above.

Recommendation #5: Additional modifications need to be made to the query Interface of the architectural data base in PAWS. The specifics of these modifications cannot be defined until the PPROTO designers decide on an enhanced data set for architectural characterization. They may consider redesigning their architectural system to have a general external interface so that any new systems can be brought in, not just PAWS data.

4.0 Summary

PPROTO capabilities have been studied and are well understood by the SU Team. Data elements from the PAWS architectural data base have been identified that can be used to define the characterization of the MultiMax and the Touchstone parallel architectures within the existing capabilities of PPROTO. This characterization is restricted by the limitations of the present version of PPROTO. The creation of data members for the PPROTO architectural characterization can be automated, but they cannot be linked to the simulation and visualization.

An attempt to establish an interface between the architectural data base of PAWS and PPROTO was completed. The objective of this work was to enhance the architectural characterization of PPROTO. An automated interface is not possible with the present form of PPROTO.

The following tasks were completed in this effort:

1. Design, test and refine the Architectural Visualization Interface to PPROTO.

2. Design, test and refine the Interface to PPROTO.

3. Design, test and revise refinement to PAWS architectural data base.

Several specific recommendations have been presented that can be utilized in order to allow PPROTO to more accurately represent parallel architectures. A discussion of the impact on PPROTO for each specific recommendation has been included. These specific recommendations and the impact on PPROTO are presented below in their order of importance:

- Provide the capability in PPROTO to use floating point values for architectural data. This solves the scaling problem and gives more meaning to the results generated by PPROTO.

  Impact on PPROTO: This is a minor programming change and a revision in output display. However, it is a major impact on the execution of their simulation. If it in fact is used to include actual timing information the result will be more meaningful because PPROTO will more closely approximate real time. It is important to recognize that this will move PPROTO toward becoming a timing performance predictor which is much more controversial than its present form.

- Provide the capability in PPROTO to model hierarchical memories. Memory and communication penalties are very critical to parallel system performance and should be modeled more accurately in PPROTO.

  Impact on PPROTO: This will require a major programming effort and is a significant change in the modeling of an architecture within the PPROTO performance estimation software. Contention analysis has to be added which is a major undertaking. The accuracy of the resultant system will be a significant improvement. over present PPROTO predictions.

- Provide the capability in PPROTO to import architectural data from disk. This will allow new systems to be easily entered and will provide more accurate data for architectural characterization.

  Impact on PPROTO: This should actually be a minor programming activity that would allow more rapid addition of architectures. It also would make it easier for companies and customers to enter architectures.

- Provide the capability in PPROTO to identify the data elements it needs in order to specify an architecture. This will keep the actual management of the PPROTO architectural characterization within PPROTO control.

  Impact on PPROTO: This is a medium level programming activity that requires standardization of the interface to the architectural characterization portion of PPROTO. It constrains the architectural description, but also insures that it is complete rather than using hidden defaults that can be incorrect as is presently done.

- Suggest that the PPROTO developers consider using a different data base system from ONTOS. One that has less hidden parameters. This should make the inclusion of large architectures easier for the PPROTO developers.

  Impact on PPROTO: This could require a complete reprogramming of PPROTO. It should only be done if the designers of PPROTO can determine that a different data base system would be significantly better. No recommendations can be given presently as to what a suitable data base may be.

*U.S. GOVERNMENT PRINTING OFFICE: 1994-510-117-50030

MISSION
OF
ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.